Discovering Rare Itemsets

Philippe Fournier-Viger
http://www.philippe-Fournier-viger.com

Source code and datasets available in the SPMF library

Introduction

- **Pattern mining:** using algorithms to discover interesting patterns in data.
- One of the most important pattern mining task is **frequent itemset mining.**
- It consists of finding sets of values (items) that appear frequently in the data (**frequent itemsets**).
- Today, I will talk about the opposite problem of discovering **rare itemsets**.

FREQUENT ITEMSET MINING (BRIEF REVIEW)

Frequent itemset mining (频繁项集挖掘)

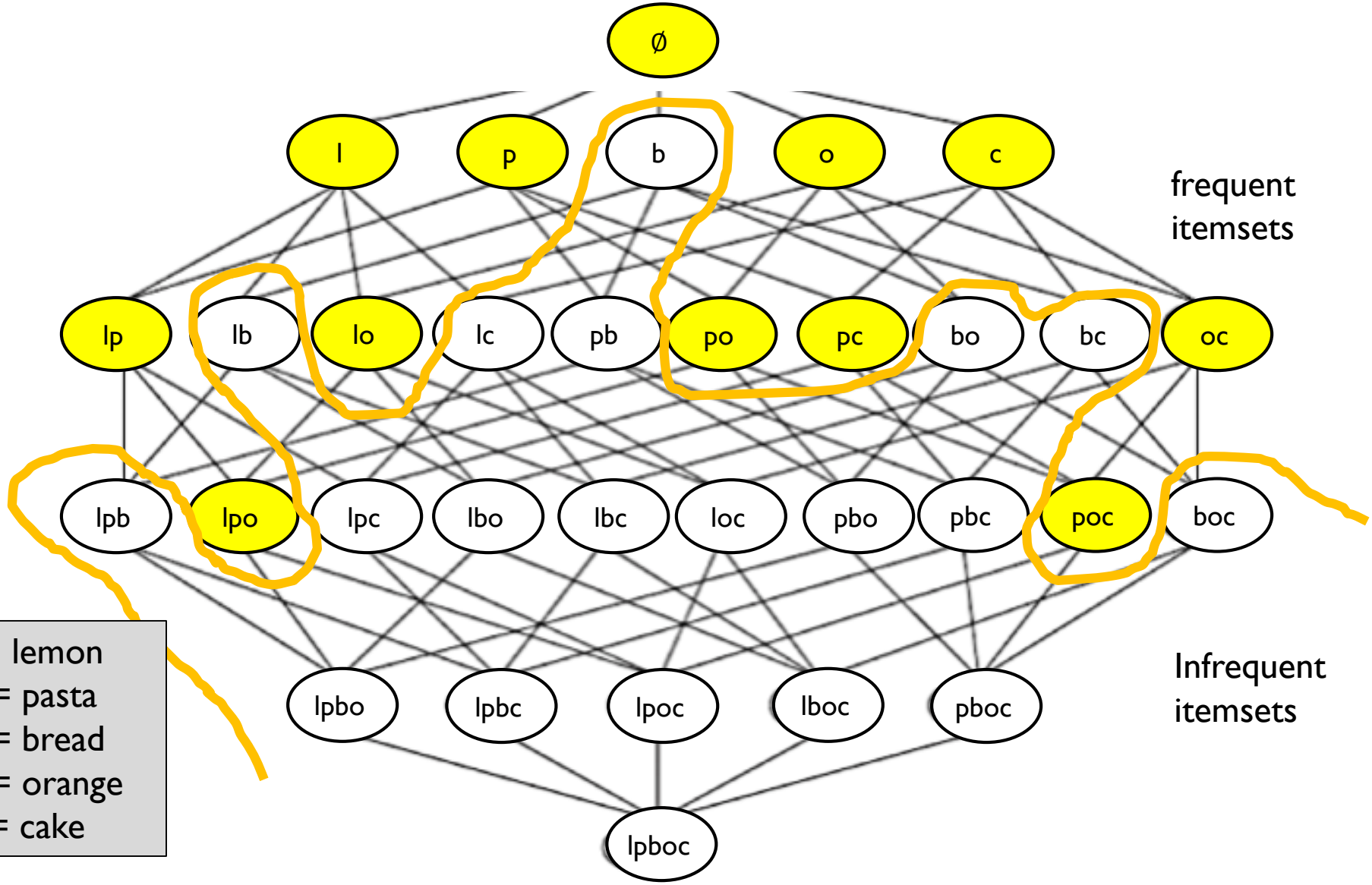
A transaction database:

Transaction	Items appearing in the transaction
T1	{pasta, lemon, bread, orange}
T2	{pasta, lemon}
T3	{pasta, orange, cake}
T4	{pasta, lemon, orange cake}

For *minsup* = 2, the frequent itemsets are:

{lemon}, {pasta}, {orange}, {cake}, {lemon, pasta}, {lemon, orange}, {pasta, orange}, {pasta, cake}, {orange, cake}, {lemon, pasta, orange}

minsup =2


Ø
l
p
b
o
c
lp
lb
lo
lc
pb
po
pc
bo
bc
oc
lpb
lpo
lpc
lbo
lbc
loc
pbo
pbc
poc
boc
lpbo
lpbc
lpoc
lboc
pboc
lpboc
frequent itemsets
Infrequent itemsets
l = lemon
p = pasta
b = bread
0 = orange
c = cake

Property 2: Let there be an itemset Y.
If there exists an itemset $X \subset Y$ such that X is infrequent, then Y is infrequent.

Example:

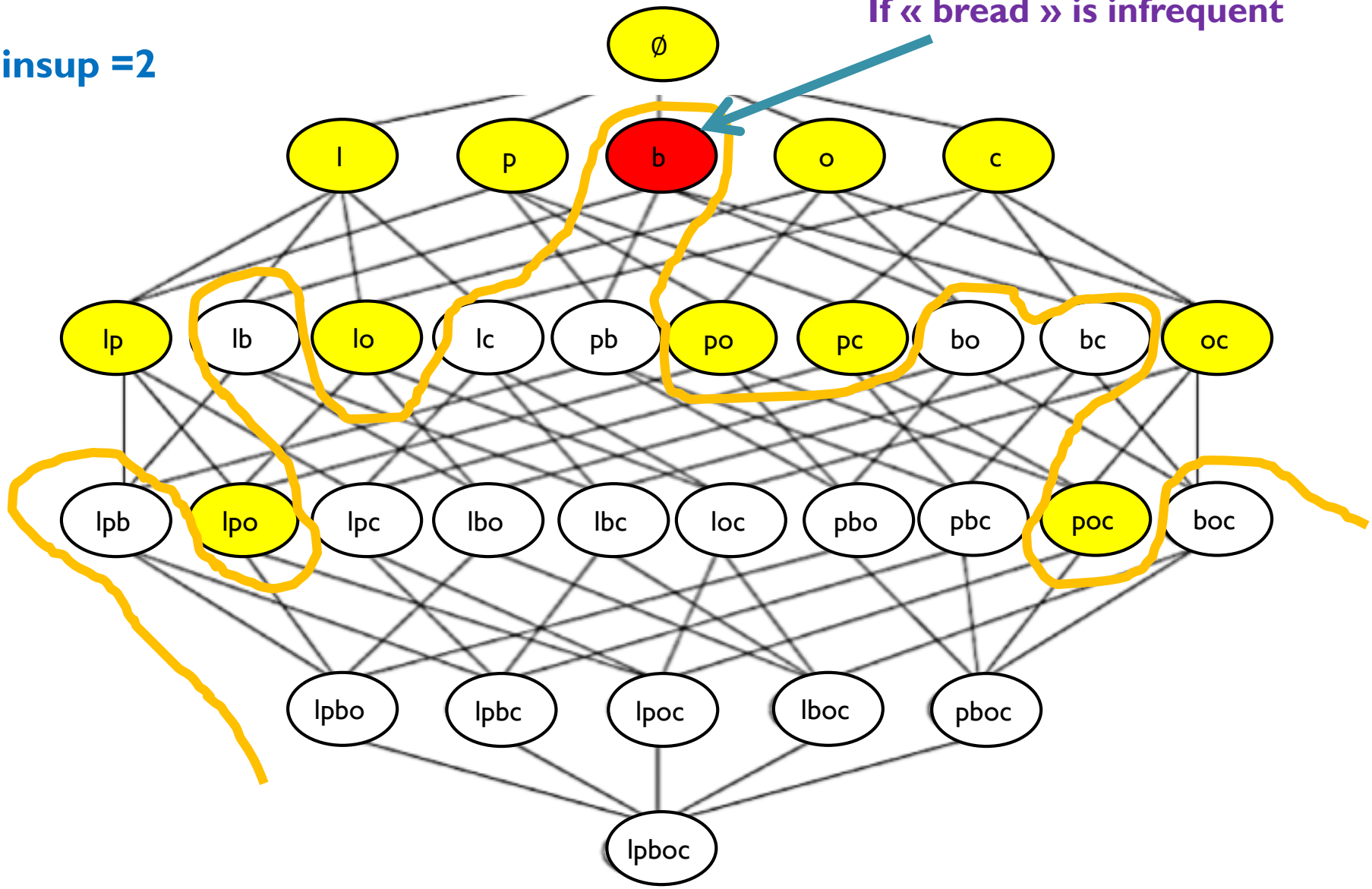
- Consider **{bread, lemon}.**
- If we know that **{bread}** is infrequent, then we can infer that **{bread, lemon}** is also infrequent.

Transaction	Items appearing in the transaction
T1	{pasta, lemon, bread, orange}
T2	{pasta, lemon}
T3	{pasta, orange, cake}
T4	{pasta, lemon, orange, cake}

This property is useful to reduce the search space.

Example:

minsup =2

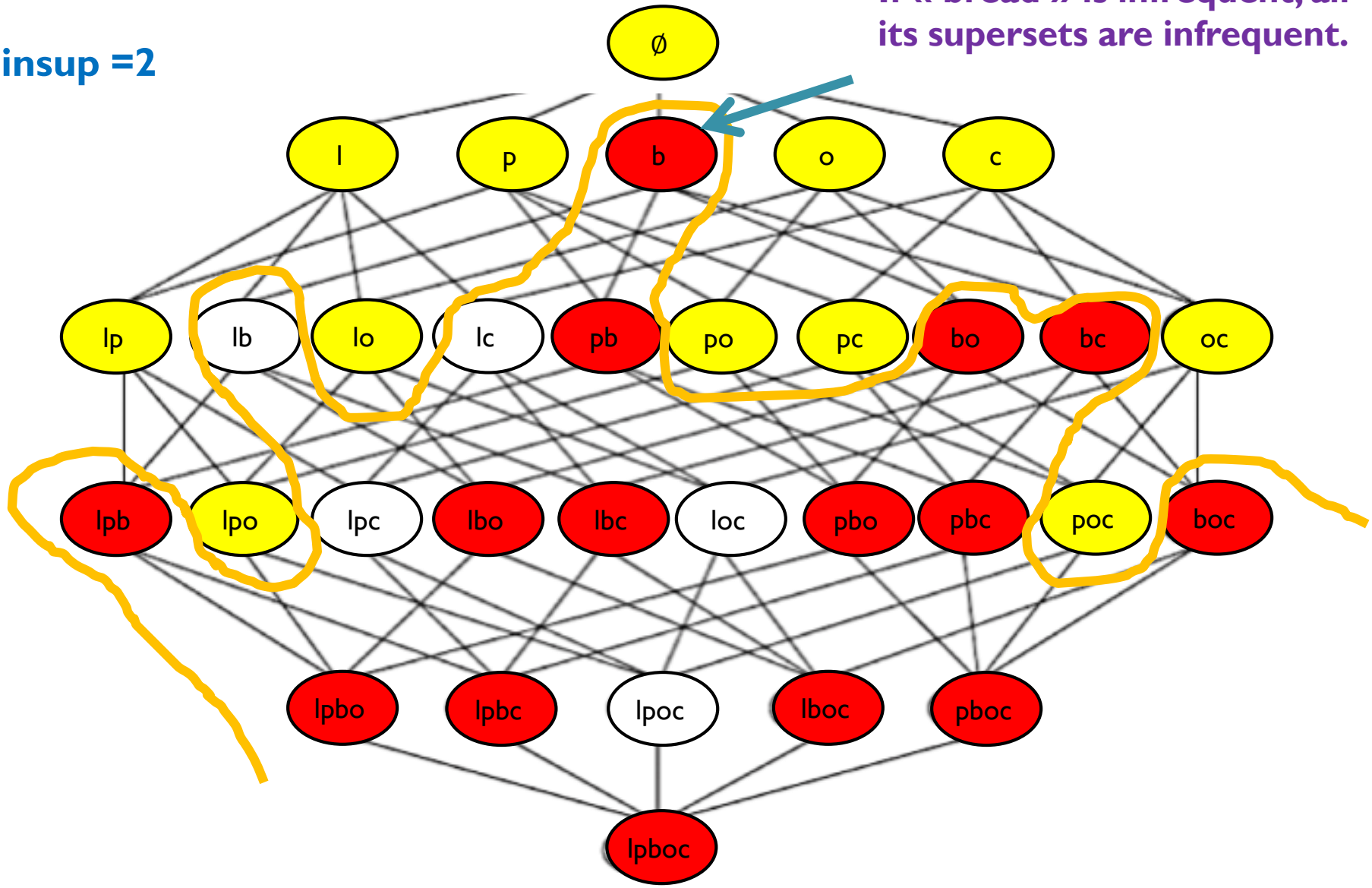


This property is useful to reduce the search space.

Example:

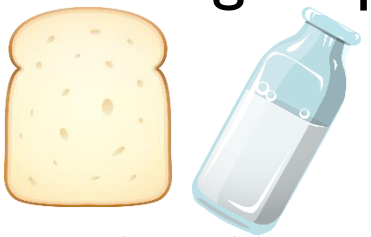
minsup =2

If « bread » is infrequent, all its supersets are infrequent.



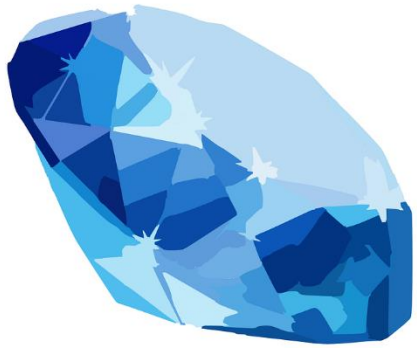
Limitations of frequent itemset mining

- There is an underlying **hypothesis** that *something frequent must be important*.
- But in practice, many frequent itemsets are unimportant.
- **Example**: many persons purchase bread and milk but it is not something surprising or profitable.



- Too many frequent patterns may make it hard to find rarer patterns that are interesting.

RARE PATTERN MINING



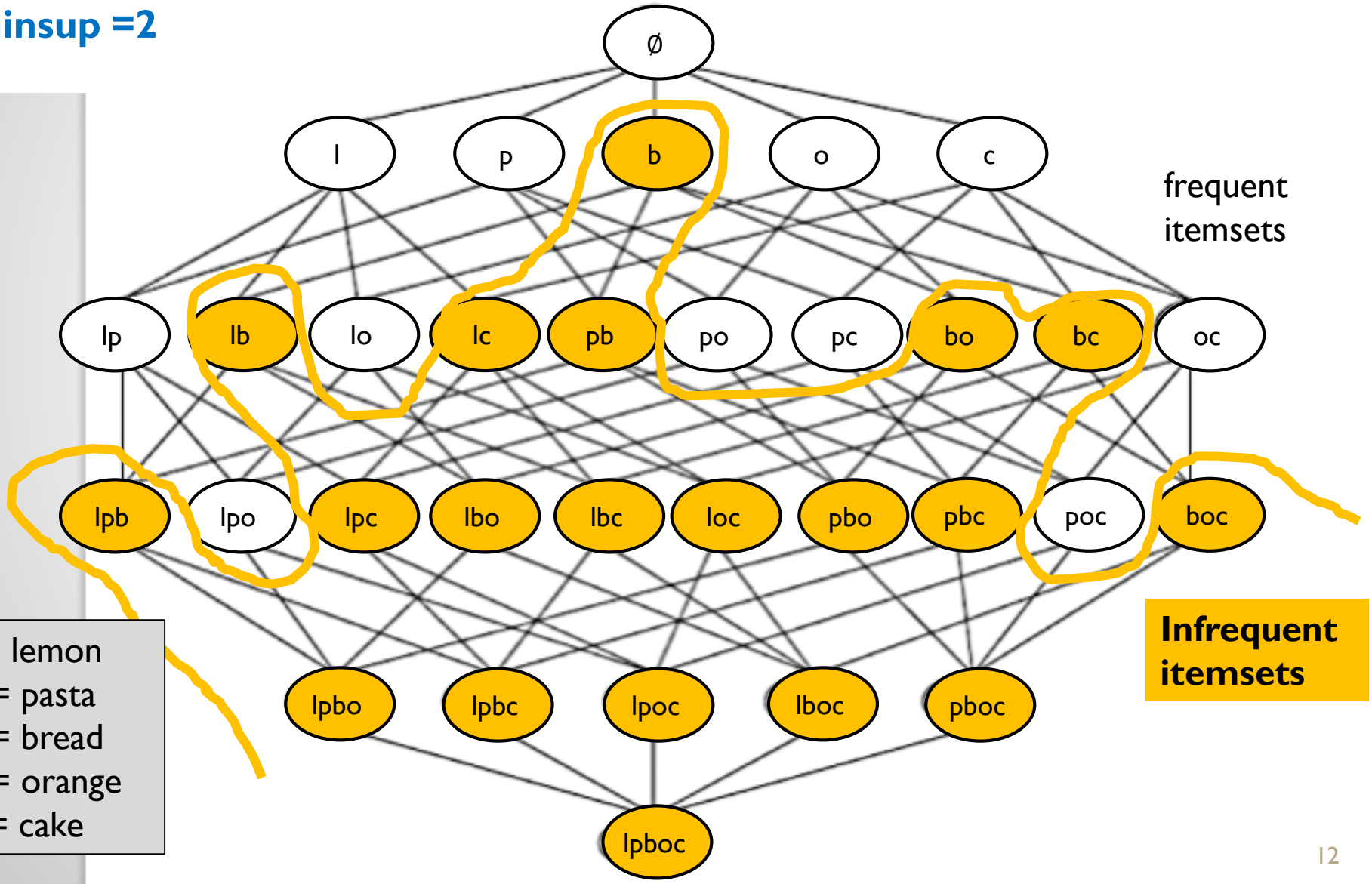
Finding rare patterns

- We first need to define what is a **rare pattern**.
- There are different definitions.
- I will give an overview.

We could define rare itemsets as infrequent itemsets

Definition 1 (infrequent itemset): An itemset X is infrequent if sup(X)<minsup.

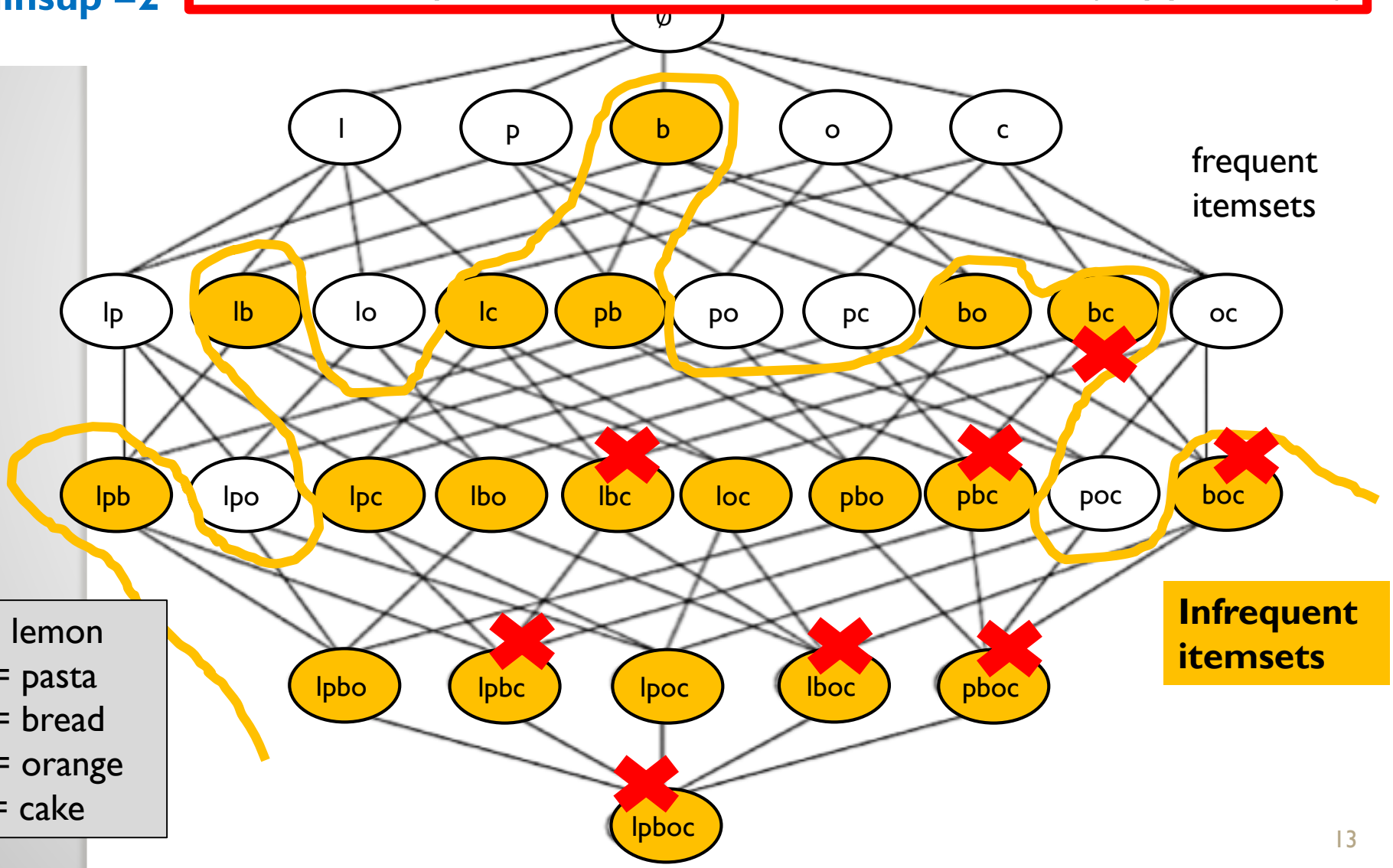
minsup =2



We could define rare itemsets as infrequent itemsets

Problem: Too many itemsets.
Some infrequent itemsets do not even exist (support = 0)

minsup =2



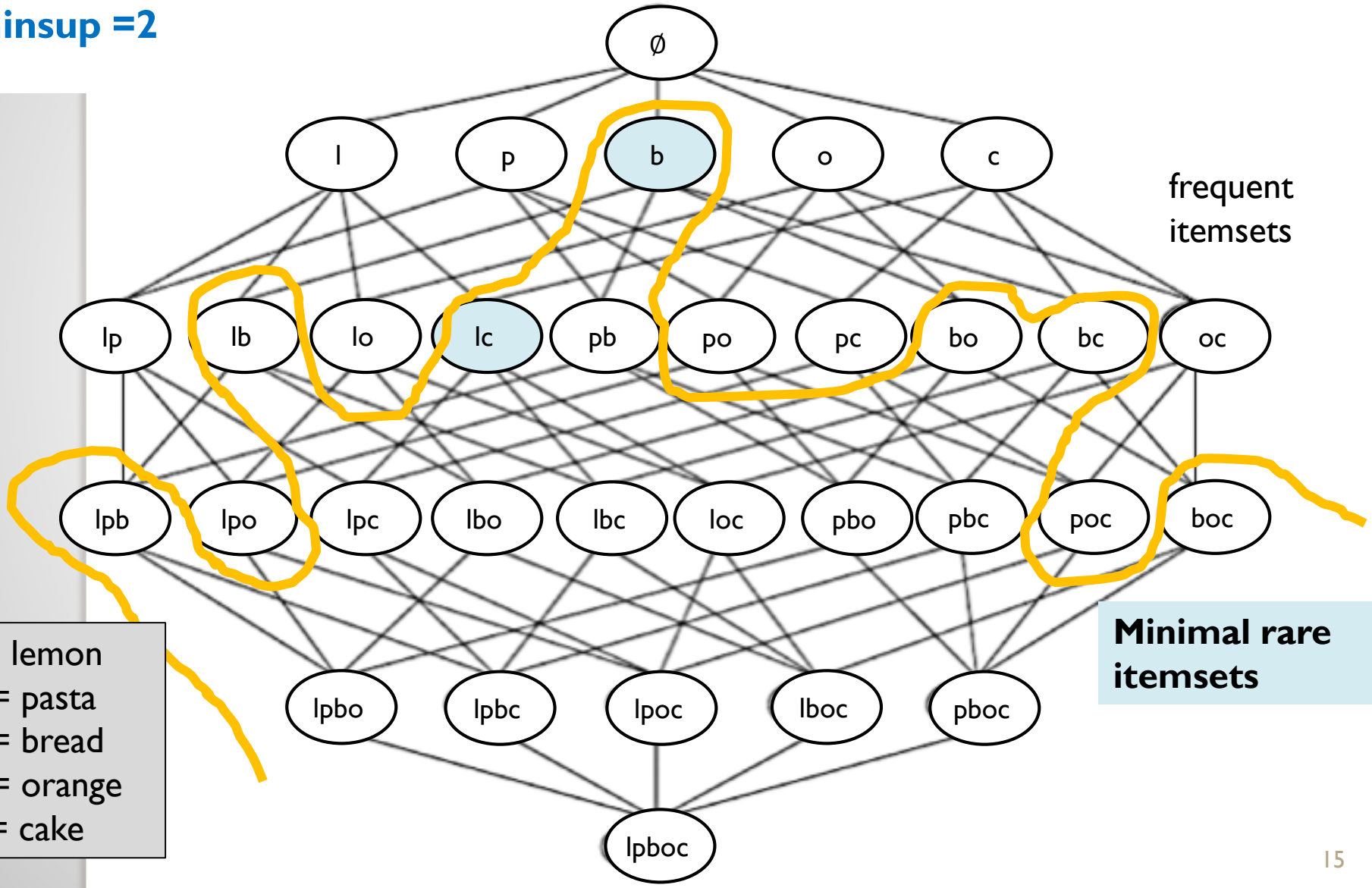
Another definition: minimal rare itemsets

*Proposed for the **AprioriRare algorithm**:*
Laszlo Szathmary, Amedeo Napoli, Petko Valtchev: Towards Rare Itemset Mining. ICTAI (1) 2007: 305-312

Definition 1 (minimal rare itemset): An itemset X is a minimal rare itemset if sup(X)<minsup and all its proper subsets are frequent itemsets (i.e. for any subset $Y \subset X$, sup(Y) $\geq$ minsup)

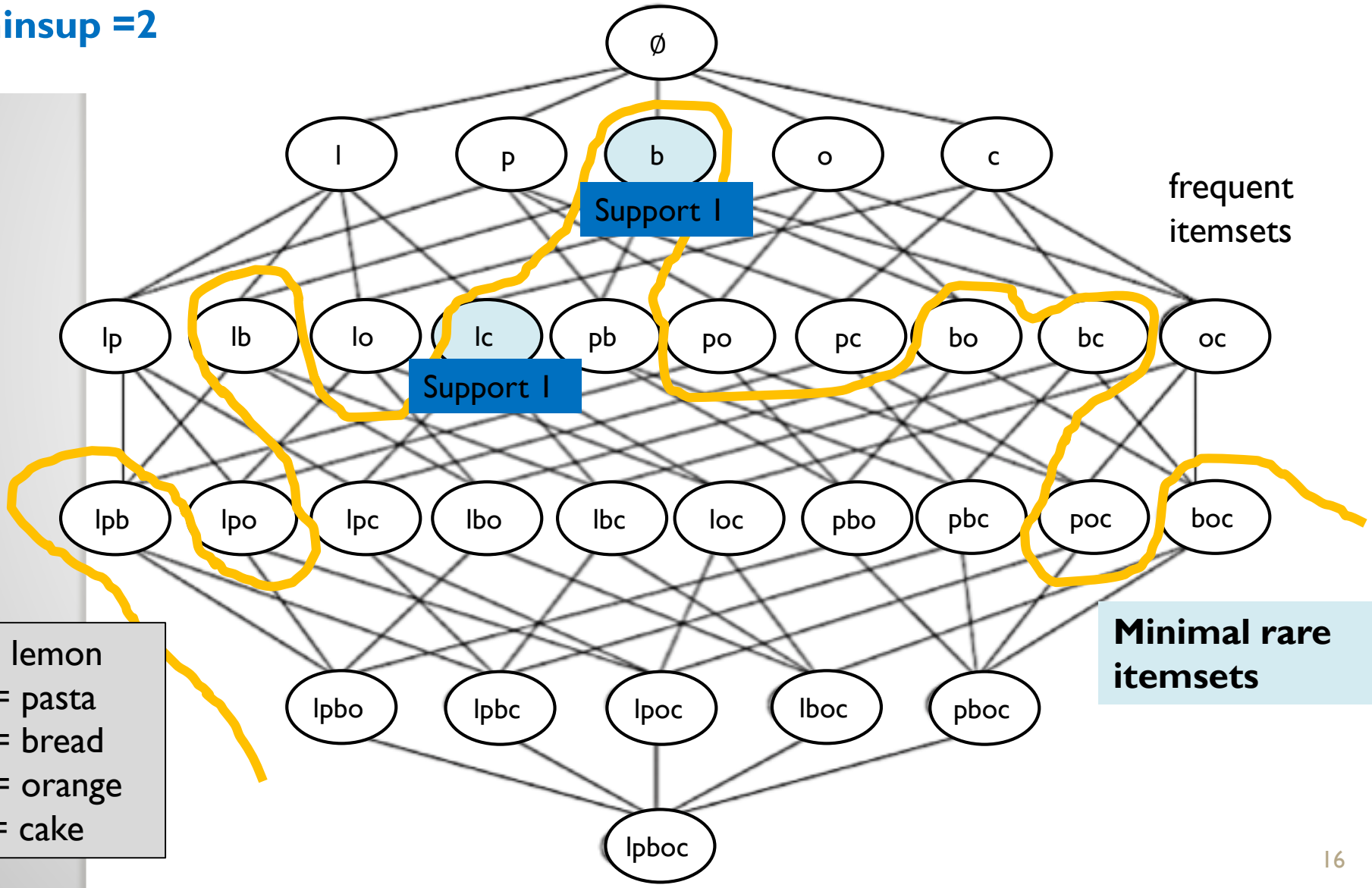
Minimal rare itemsets

minsup =2



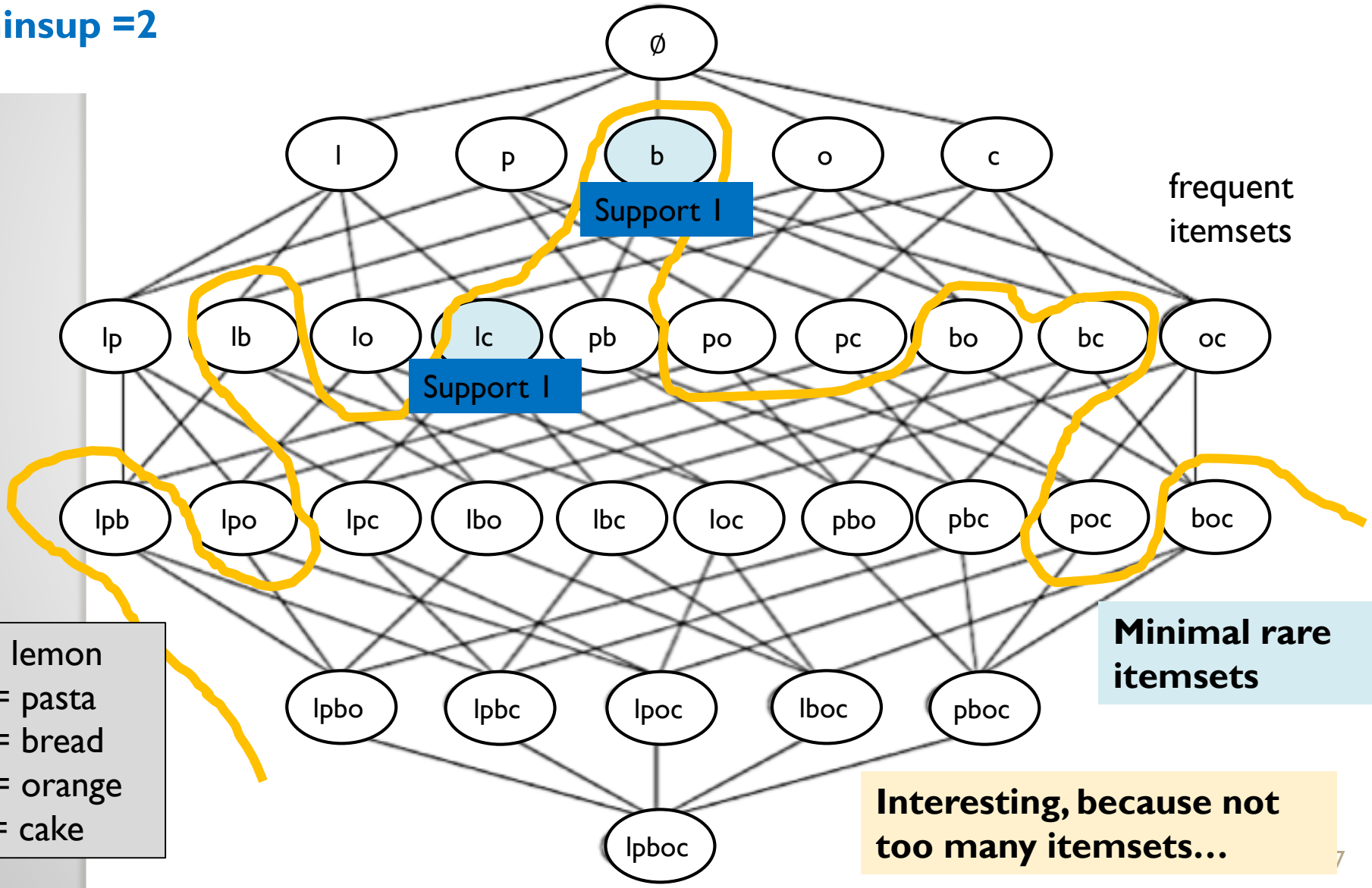
Minimal rare itemsets

minsup =2



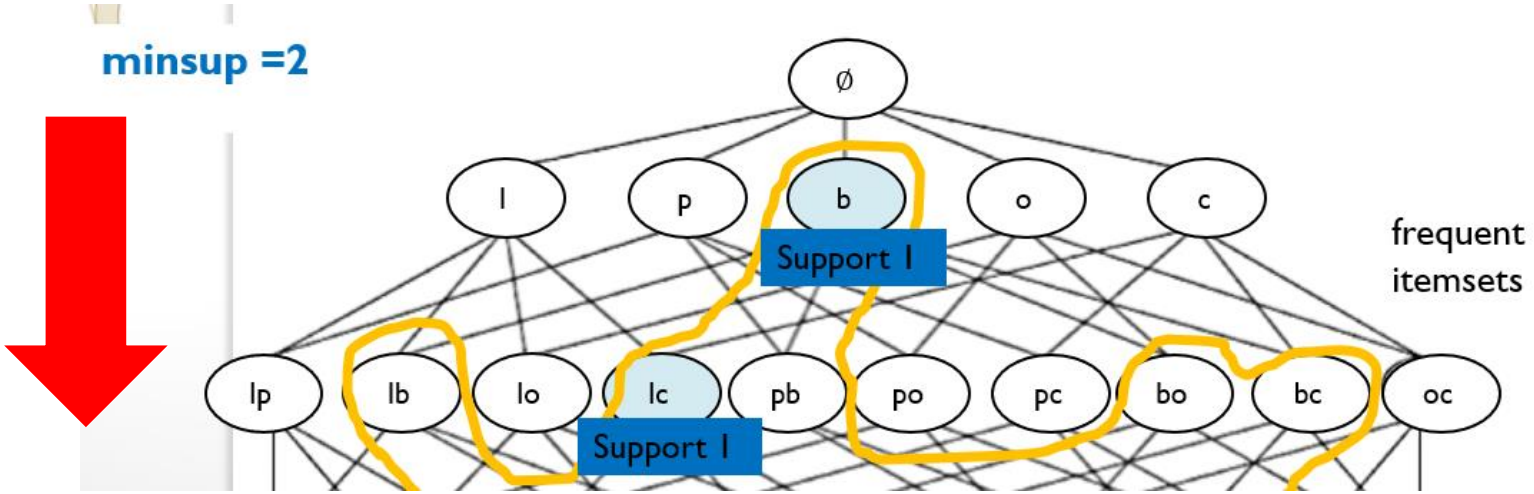
Minimal rare itemsets

minsup =2



How can we find the minimal rare itemsets?

- It is not easy!
- Generally, frequent itemset mining algorithms start from single items and combine them to find larger itemsets.
- As itemsets become larger, the support can decrease.
- Thus, to search for rare itemsets, we must « pass through » the frequent itemsets to reach the rare itemsets. How?



The AprioriRare algorithm (2007)

- It is based on Apriori
- As Apriori, the itemsets are generated by levels:
  - Itemsets of size 1 (one item)
  - Itemsets of size 2 (two items)
  - Itemsets of size 3 (three items)
  - …
- Two differences:
  - If AprioriRare finds an itemset of size k that is infrequent, AprioriRare checks if its subsets are frequent. If yes, it is a minimal rare itemset.
  - AprioriRare do not use the infrequent itemsets to generate larger itemsets.

The AprioriRare algorithm

- I will now explain how the AprioriRare algorithm works
- **Input:**
  - *minsup*
  - a transactional database
- **Output:**
  - all the minimal rare itemsets

Consider ***minsup* =2.**

The AprioriRare algorithm

Step 1: Scan the database to calculate the support of all itemsets of size 1.

e.g.

{pasta}	support = 4
{lemon}	support = 3
{bread}	support = 1
{orange}	support = 3
{cake}	support = 2

The AprioriRare algorithm

Step 2: Check all infrequent itemsets. If all subsets are frequent, they are are minimal rare itemsets.

e.g.

{pasta} support = 4
{lemon} support = 3
{bread} support = 1
{orange} support = 3
{cake} support = 2

The AprioriRare algorithm

Step 2: Check all infrequent itemsets. If all subsets are frequent, they are are minimal rare itemsets.

e.g.

{pasta} support = 4
{lemon} support = 3
{bread} support = 1 ⇦ **Minimal Rare Itemset**
{orange} support = 3
{cake} support = 2

The AprioriRare algorithm

Step 2: Keep frequent itemsets.

e.g.

{pasta}	support = 4
{lemon}	support = 3
{orange}	support = 3
{cake}	support = 2

The AprioriRare algorithm

Step 3: Generate candidates of size 2 by combining pairs of frequent itemsets of size 1.

Frequent items

{pasta}
{lemon}
{orange}
{cake}

Candidates of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{lemon, cake}
{orange, cake}

The AprioriRare algorithm

Step 5: Scan the database to calculate the support of remaining candidate itemsets of size 2.

Candidates of size 2

{pasta, lemon} support: 3

{pasta, orange} support: 3

{pasta, cake} support: 2

{lemon, orange} support: 2

{lemon, cake} support: 1

{orange, cake} support: 2

The AprioriRare algorithm

Step 6: For each infrequent itemset, check if all the subsets are frequent

Candidates of size 2

{pasta, lemon} support: 3

{pasta, orange} support: 3

{pasta, cake} support: 2

{lemon, orange} support: 2

{lemon, cake} support: 1

{orange, cake} support: 2

The AprioriRare algorithm

Step 6: For each infrequent itemset, check if all the subsets are frequent

Candidates of size 2

{pasta, lemon} support: 3

{pasta, orange} support: 3

{pasta, cake} support: 2

{lemon, orange} support: 2

{lemon, cake} support: 1 ⟸ **Minimal Rare Itemset**

{orange, cake} support: 2

The AprioriRare algorithm

Step 6: Keep frequent itemsets of size 2

Frequent itemsets of size 2

{pasta, lemon} support: 3

{pasta, orange} support: 3

{pasta, cake} support: 2

{lemon, orange} support: 2

{orange, cake} support: 2

The AprioriRare algorithm

Step 7: Generate candidates of size 3 by combining frequent pairs of itemsets of size 2.

Frequent itemsets of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{orange, cake}

Candidates of size 3

{pasta, lemon, orange}
{pasta, lemon, cake}
{pasta, orange, cake}
{lemon, orange, cake}

The AprioriRare algorithm

Step 7: Scan the database to find the support of candidates

Frequent itemsets of size 2

{pasta, lemon}
{pasta, orange}
{pasta, cake}
{lemon, orange}
{orange, cake}

Candidates of size 3

{pasta, lemon, orange}: 2
{pasta, lemon, cake}: 1
{pasta, orange, cake}:2
{lemon, orange, cake}: 1

The AprioriRare algorithm

Step 8: For each infrequent itemset, check if all the subsets are frequent

Frequent itemsets of size 2

{pasta, lemon}

{pasta, orange}

{pasta, cake}

{lemon, orange}

{orange, cake}

Candidates of size 3

{pasta, lemon, orange}:2

{pasta, lemon, cake}: 1

{pasta, orange, cake}:2

{lemon, orange, cake}: 1

The AprioriRare algorithm

Step 8: For each infrequent itemset, check if all the subsets are frequent

Frequent itemsets of size 2

{pasta, lemon}

{pasta, orange}

{pasta, cake}

{lemon, orange}

{orange, cake}

Candidates of size 3

{pasta, lemon, orange}:2

{pasta, lemon, cake}: 1

{pasta, orange, cake}:2

{lemon, orange, cake}: 1

The AprioriRare algorithm

Step 8: For each infrequent itemset, check if all the subsets are frequent

Frequent itemsets of size 2

{pasta, lemon}

{pasta, orange}

{pasta, cake}

{lemon, orange}

{orange, cake}

Candidates of size 3

{pasta, lemon, orange}

{pasta, orange, cake}

The AprioriRare algorithm

Step 10: Keep the frequent itemsets (all)

frequent itemsets of size 3

{pasta, lemon, orange} support: 2

{pasta, orange, cake} support: 2

The AprioriRare algorithm

Step 11: generate candidates of size 4 by combining pairs of frequent itemsets of size 3.

Frequent itemsets of size 3

{pasta, lemon, orange}

{pasta, orange, cake}

Candidates of size 4

{pasta, lemon, orange, cake}

The AprioriRare algorithm

Step 12: Check to see if the subsets of each infrequent itemset are frequent. They are not.

Frequent itemsets of size 3

{pasta, lemon, orange}

{pasta, orange, cake}

Candidates of size 4

The AprioriRare algorithm

Step 12: Since there is no more frequent itemsets, we cannot generate candidates of size 5 and the algorithm stops.

Candidates of size 4

~~{pasta, lemon, orange, cake}~~

Result →

Final result

The minimal rare itemsets:

{bread}	support = 1
{lemon, cake}	support = 1

Another definition: perfectly rare itemsets

*Proposed for the **AprioriInverse** algorithm:*
Yun Sing Koh, Nathan Rountree: Finding Sporadic Rules Using Apriori-Inverse. PAKDD 2005: 97-106

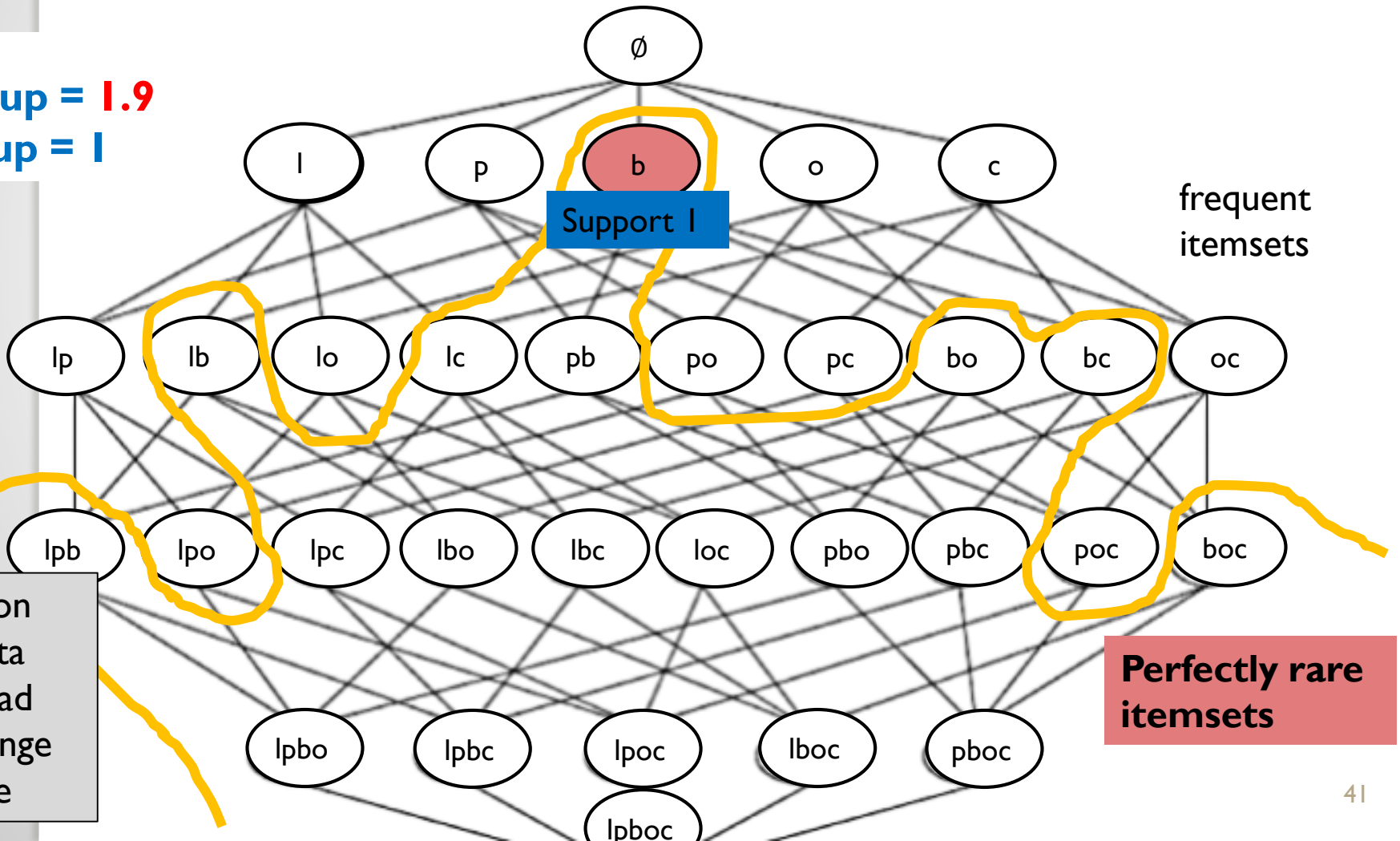
Definition 1 (perfectly rare itemset):
Let there be two thresholds minsup and maxsup, such that maxsup > minsup.

An itemset Z is a **frequent itemset** if sup(Z)≥maxsup.

An itemset X is a **perfectly rare itemset**
if sup(X)≥ minsup and sup(X)<maxsup
and for any non empty subset $Y \subset X$, sup(Y) < maxsup.

Definition 1 (perfectly rare itemset):
Let there be two thresholds minsup and maxsup, such that maxsup > minsup.
An itemset Z is a **frequent itemset** if sup(Z)≥maxsup.
An itemset X is a **perfectly rare itemset** if sup(X)≥ minsup,
sup(X)<maxsup and for any non empty subset $Y \subset X$, sup(Y) < maxsup.

maxsup = 1.9
minsup = 1



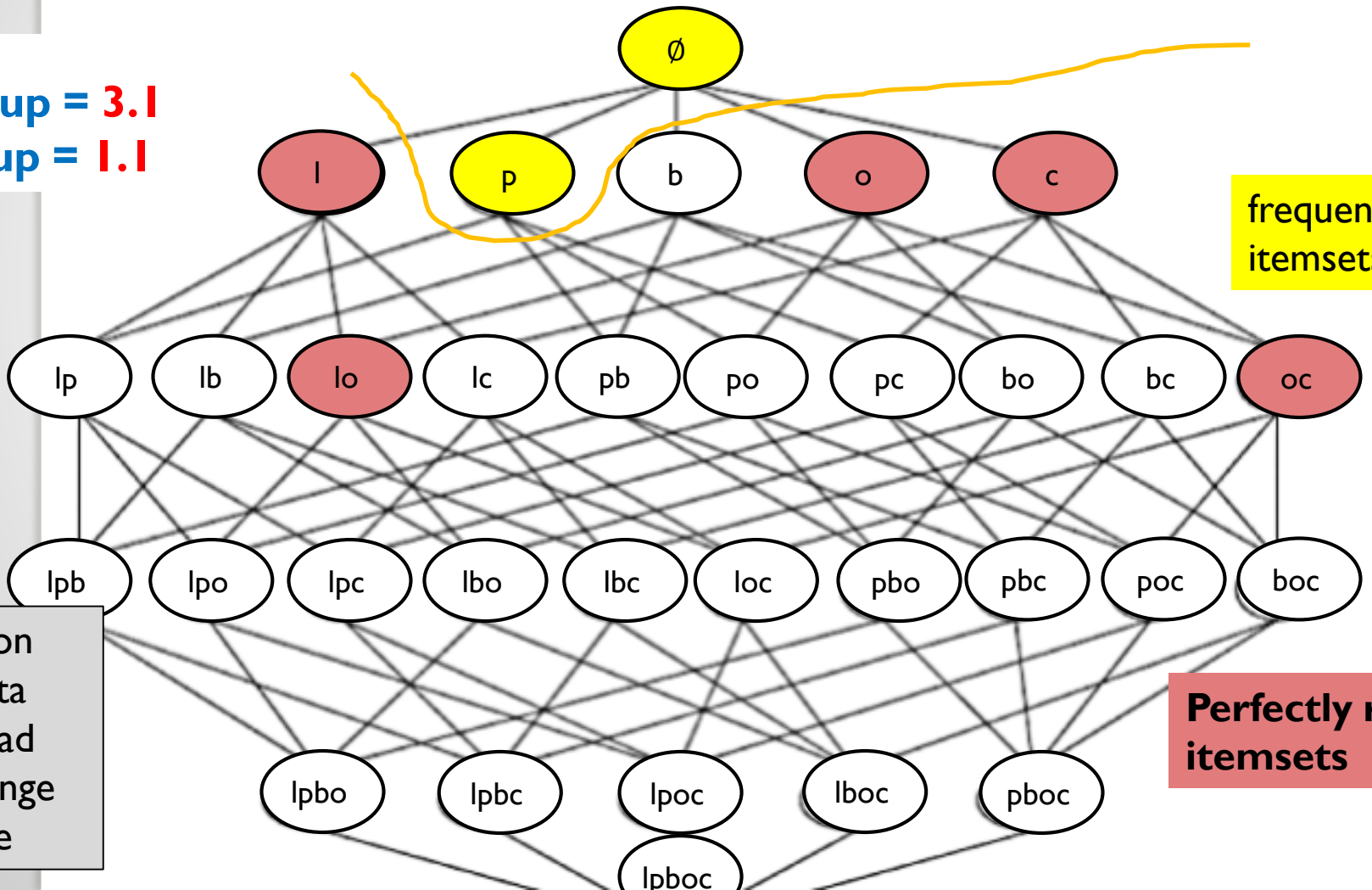
frequent itemsets

Perfectly rare itemsets

l = lemon
p = pasta
b = bread
0 = orange
c = cake

Definition 1 (perfectly rare itemset):
Let there be two thresholds minsup and maxsup, such that maxsup > minsup.
An itemset Z is a **frequent itemset** if sup(Z)≥maxsup.
An itemset X is a **perfectly rare itemset** if sup(X)≥ minsup,
sup(X)<maxsup and for any non empty subset $Y \subset X$, sup(Y) < maxsup.

maxsup = 3.1
minsup = 1.1



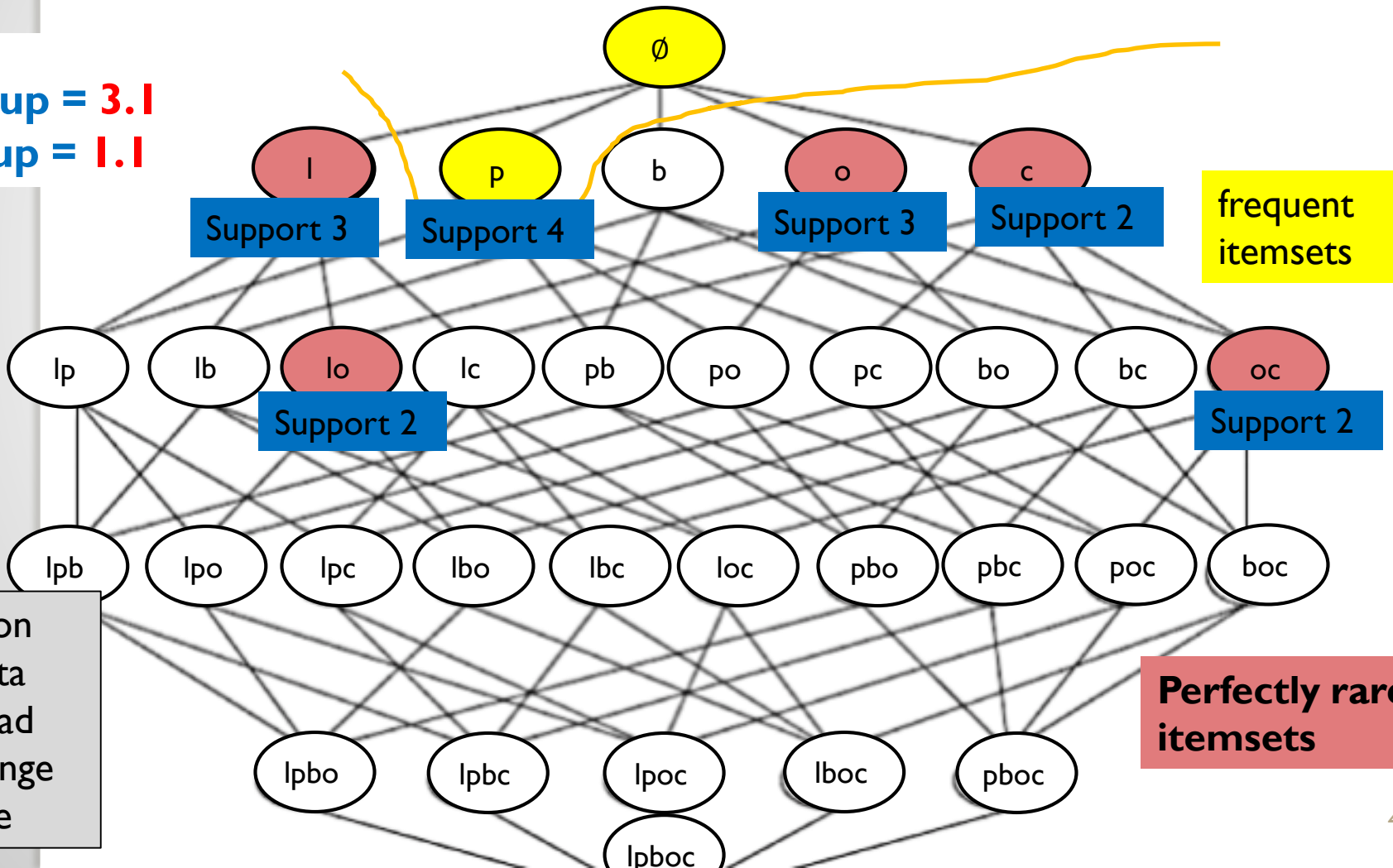
frequent itemsets

Perfectly rare itemsets

l = lemon
p = pasta
b = bread
0 = orange
c = cake

Definition 1 (perfectly rare itemset):
Let there be two thresholds minsup and maxsup, such that maxsup > minsup.
An itemset Z is a **frequent itemset** if sup(Z)≥maxsup.
An itemset X is a **perfectly rare itemset** if sup(X)≥ minsup,
sup(X)<maxsup and for any non empty subset $Y \subset X$, sup(Y) < maxsup.

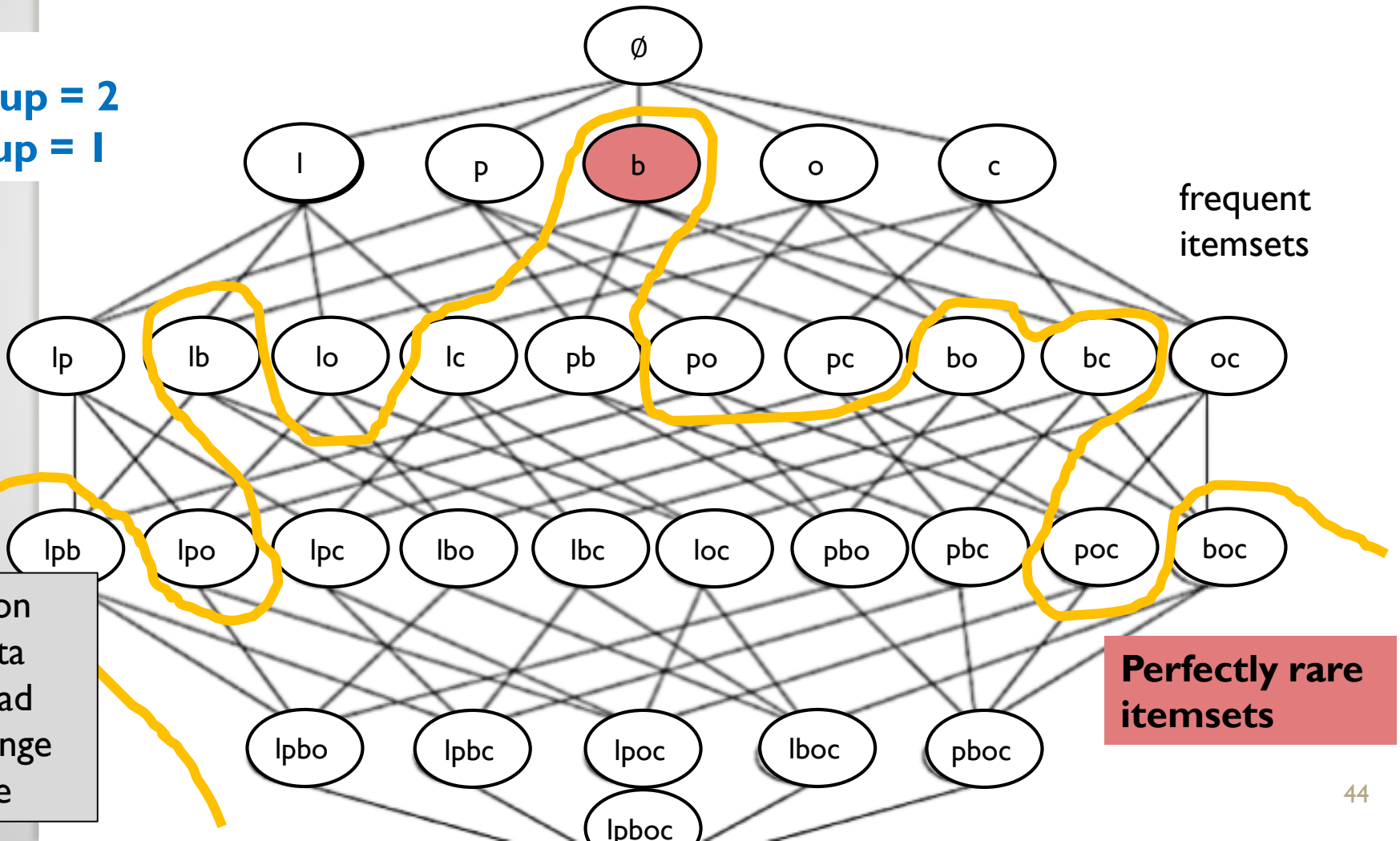
maxsup = 3.1
minsup = 1.1



l = lemon
p = pasta
b = bread
0 = orange
c = cake

Definition 1 (perfectly rare itemset):
Let there be two thresholds minsup and maxsup, such that maxsup > minsup.
An itemset Z is a **frequent itemset** if sup(Z)≥maxsup.
An itemset X is a **perfectly rare itemset** if sup(X)≥ minsup, sup(X)<maxsup and for any non empty subset $Y \subset X$, sup(Y)<maxsup.

maxsup = 2
minsup = 1



frequent itemsets

l = lemon
p = pasta
b = bread
0 = orange
c = cake

Perfectly rare itemsets

How to find perfectly rare itemsets?

- AprioriInverse (2005)
- Based on Apriori.
- Key difference:
  - Initially, AprioriInverse discards each item **x** such that sup(x) ≥ maxsup because we don't need such item.
  - After that AprioriInverse search for itemsets using the remaining items just like Apriori to find itemsets with a support no less than *minsup*.

Conclusion

This video has presented:

- The problem of **rare itemset mining**
- **Three definitions of rare itemsets:**
  - Infrequent itemsets
  - Minimal rare itemsets
  - Perfectly rare itemsets
- **Two algorithms:**
  - AprioriRare
  - AprioriInverse
- To find rare itemsets that are more interesting, we can also combine the concept of rare itemsets with that of correlated itemset (e.g. the CORI algorithm).